OKTICKET S.L.

API - Funcionalidades extendidas y ejemplos

1 OBJETIVO Y ALCANCE

El presente documento presenta diversos ejemplos y casos de uso avanzados de la API de Okticket, previamente definida en el documento Guía de Uso.

Los ejemplos están orientados a aquellos desarrolladores que ya hayan tenido una primera toma de contacto con la API y conozcan los detalles básicos de la misma: autenticación, paginación, etc, y que necesiten de funcionalidades extendidas de Okticket.

2 EJEMPLOS

Se agrupan los ejemplos mostrados por bloque funcional o entidad.

1. Gestión de campos de tipo Lista externa

En Okticket existe la posibilidad de activar y configurar nuevos campos personalizables por cliente. Estos campos pueden activarse a nivel de Gasto u Hoja de gastos, y pueden ser distintos en función del tipo de gasto, la empresa, la categoría del gasto o el rol de usuario.

Dentro de los campos personalizable disponibles en Okticket, existen el tipo de campo **Lista Externa**, que permite que sus valores se obtengan de una tabla alimentada vía API.

El campo aparece en la APP y el gestor de Okticket como un selector (que puede ser múltiple), que muestra una lista de textos (etiqueta) y que guarda un valor al ser seleccionado, en el gasto u hoja correspondiente. Este valor puede luego recuperarse dentro de la entidad correspondiente, en el atributo `custom_fields.<id_campo>`.

Cada elemento de la lista se compondrá de cuatro atributos principales:

- `label`: Etiqueta, lo que se muestra a los usuarios en pantalla
- `value`: Valor, lo que se guardar en el modelo
- `global`: Si es global, el elemento le aparecerá a todos los usuarios; si no, solo a los que se configuren para ello.
- `active`: Si está activo, se mostrará en la lista.

Se muestran a continuación los ejemplos de llamadas necesarias para alimentar, configurar y mantener los elementos de un campo de tipo **Lista externa** llamado **`project`**.

- Para consultar los valores que hay disponibles para ese campo:
    - GET **/api/external-lists/project**
- Para consultar los valores que hay disponibles para ese campo, junto con los usuarios asociados a dichos valores:

- GET **/api/external-lists/project?with=users**

- Para consultar los valores que tiene disponibles un usuario concreto con ID=<uid>
  - GET **/api/external-lists/project?user=<uid>**

- Para añadir un nuevo elemento a la lista:
  - Global: lo verán todos los usuarios:
    - POST **/api/external-lists/project**
    - ```
      {
            "active": true,
            "global": true,
            "label": "Parte 01",
            "value": "PT202021201",
            "field": "project",
            "company_id": {{companyId}}
      }
      ```
  - A un usuario o usuarios particulares:
    - POST **/api/external-lists/project**
    - ```
      {
            "active": true,
            "global": false,
            "label": "Equinox Fase 2",
            "value": "EQPT202020213",
            "field": "project",
            "company_id": {{companyId}},
            "ids_users": [{{idUser1}},{{idUser2}}]
      }
      ```

- Para desactivar un elemento y ya no pueda usarse:
  - PATCH **/api/external-lists/project/{{idValor}}**
  - ```
    {
          "active": false
    }
    ```

- Para vincular un elemento ya existente a un o varios usuarios:
  - POST **/api/external-lists/project/{{idValor}}/actions/attach-users**
  - ```
    {
          "ids_users": [{{idUser1}},{{idUser2}}]
    }
    ```

- Para desvincular un elemento ya existente de un usuario:
  - POST **/api/external-lists/project/{{idValor}}/actions/detach-users**
  - ```
    {
          "ids_users": [{{idUser3}}]
    }
    ```
    ```

2. Gestión de Grupos de validación y Workflows

1. Introducción

Para entender correctamente este apartado es necesario tener claros ciertos conceptos manejados en el proceso de validación de un hoja de gastos dentro de Okticket:

1. **Estados**: Situación actual de una hoja de gastos que típicamente tendrá los valores de *Abierta, Pendiente de de Revisión, En Revisión, Aprobada, Rechazada y Tramitada.*
2. **Transiciones**: Movimiento o acción de cambiar una hoja de un estado a otro.
3. **Responsable de transición**: Como indica su nombre es el responsable de realizar la transición de un estado a otro. Puede haber varios por cada transición, y pueden ser Roles o Usuarios.
4. **Flujo de aprobación o validación (workflow o WF):** Concepto que engloba los pasos a seguir para llevar a cabo una validación. Se compone de estados, transiciones y responsables de estos mismos.
5. **Grupos de validación (*departamentos*):** Agrupación de usuarios cuyas hojas de gasto tienen los mismos pasos y responsables de validación, es decir, el mismo flujo de validación.

Dado que un **grupo de validación** se utiliza para agrupar a los usuarios cuyas hojas de gasto tienen el mismo flujo de validación, en cada grupo deben estar incluidos tanto los usuarios propietarios de la hoja de gastos a validar, cómo los distintos usuarios validadores que participen en el proceso.

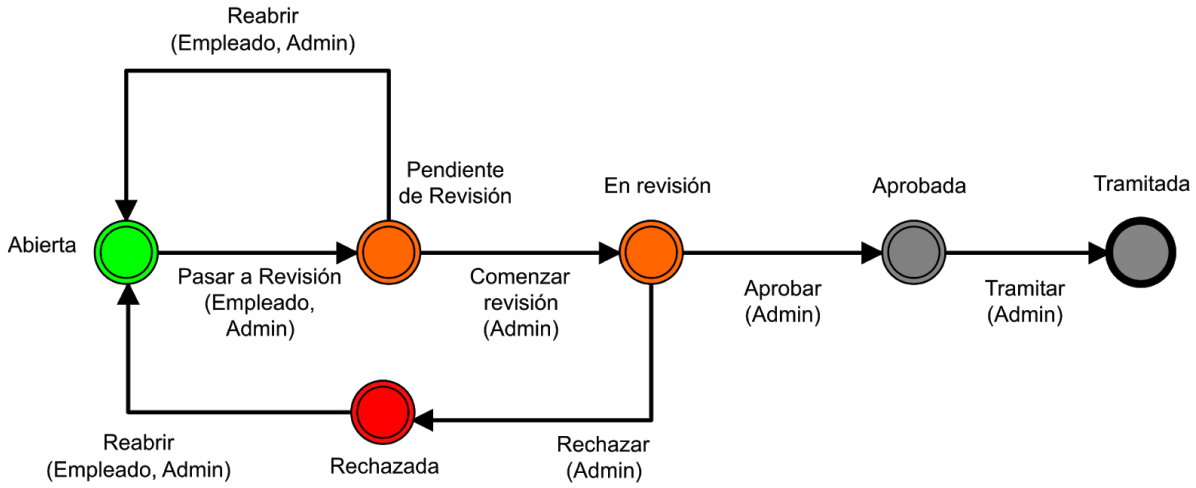
Los pasos y los responsables de las transiciones entre estados quedan definidos por el Workflow que debe estar asociado a cada grupo.

En Okticket hay ciertos WFs predefinidos que pueden usarse, aunque también pueden diseñarse nuevos para adaptarse a los procesos de validación de los distintos clientes.

Se muestran a continuación un par de ejemplos basado en los WF por defecto de Okticket:

1. Workflows

2. WF por defecto

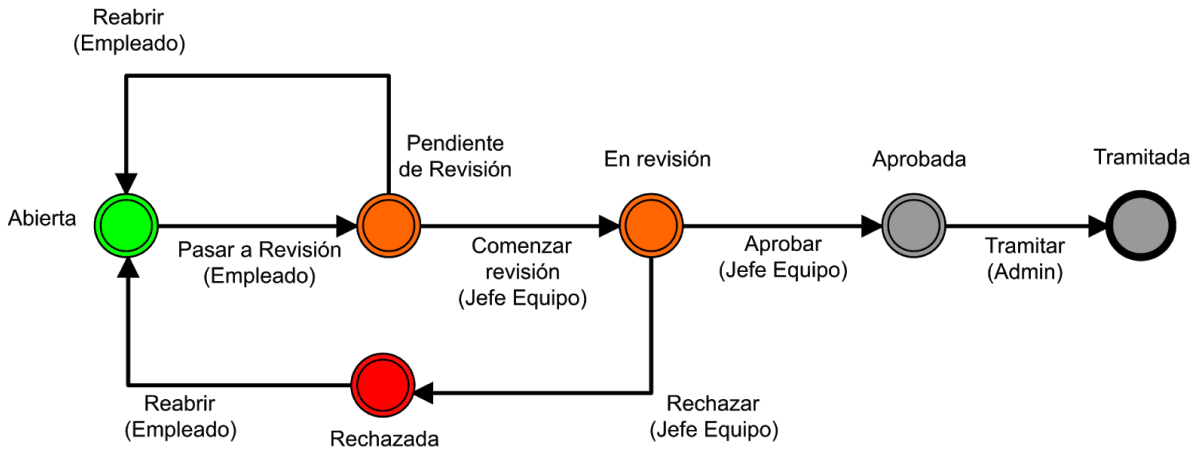


Roles: Empleado (ID: 3) y Administrador (ID: 2)
Uso: Para aquellos clientes en los que solo hay un validador o un solo paso de validación.
Modalidades:

1. Un validador único para toda la empresa: no hace falta crear grupos de validación, todos los usuarios tienen el mismo flujo a nivel de empresa y el validador es siempre el mismo: un administrador a nivel de empresa.
2. Un validador por grupo: Se añaden a cada grupo todos los usuarios como empleados y su validador como Administrador del grupo.

Para este ejemplo, solo aplicaría el segundo caso, ya que es el único en que se usan grupos de validación.

2.2.1.3 WF con jefes de equipo



Roles: Empleado (3), Jefe de Equipo (6) y Administrador (2).
Uso: Para aquellos clientes en los que solo haya **dos pasos de validación**.
Modalidades:

1. Un primer validador por grupo, y un segundo validador único para toda la empresa: Se añaden a cada grupo todos los usuarios del mismo como Empleados y a su validador como Jefe de equipo. El segundo validador es Administrador a nivel de empresa, no hace falta asociarlo a cada grupo.
2. Dos validadores distintos por grupo: Se añaden a cada grupo todos los usuarios del mismo como Empleados, el primer validador como Jefe de equipo, y el segundo validador como Administrador.

2. Gestión de grupos de validación

1. Crear nuevo grupo

- POST **/api/departments**

```
{
      "company_id": <id_empresa>,
      "name": "Grupo proyecto Alfa"
}
```

- Devuelve un JSON con los datos del grupo creados

```
{
    "data": {
        "id": 1295
        "company_id": <id_empresa>,
        "name": "Grupo proyecto Alfa",
        "updated_at": "2020-04-24 11:44:32",
        "created_at": "2020-04-24 11:44:32",

    },
    "status": "ok"
}
```

2. Añadir empleado a un grupo

Para añadir el usuario <id_usuario> como Empleado del grupo <id_grupo>, hay que realizar la siguiente llamada:

- PATCH **/api/users/<id_usuario>**

```
{
    "ids_departments": {
       "<id_grupo>": {
```

```
            "id_role": 3,
            "web_access": 1,
            "app_access": 1
        }
    }
}
```

- El parámetro APP_ACCESS debe ir a 1para permitir imputar gastos al usuario en el grupo desde la APP.
- Un mismo usuario puede pertenecer a varios grupos de validación, en cuyo caso, deberá elegir grupo al crear los gastos desde la APP.

3. Añadir administrador a un grupo

Para añadir el usuario <id_usuario> como Administrador del grupo <id_grupo>, hay que realizar la siguiente llamada:

- PATCH **/api/users/<id_usuario>**

```
{
    "ids_departments": {
        "<id_grupo>": {
            "id_role": 2,
            "web_access": 1,
            "app_access": 0
        }
    }
}
```

- El parámetro APP_ACCESS debe ir a 0 ya que los validadores no imputan gastos en dicho grupo desde la APP.
- Puede haber más de un administrador por grupo. Cualquiera de ellos podrá realizar las validaciones.

4. Añadir Jefe de equipo a un grupo

Para añadir el usuario <id_usuario> como Jefe de Equipo del grupo <id_grupo>, hay que realizar la siguiente llamada:

- PATCH **/api/users/<id_usuario>**

```
{
    "ids_departments": {
        "<id_grupo>": {
            "id_role": 6,
            "web_access": 1,
            "app_access": 0
        }
    }
}
```

- El parámetro APP_ACCESS debe ir a 0 ya que los validadores no imputan gastos en dicho grupo desde la APP.
- Puede haber más de un jefe de equipo por grupo. Cualquiera de ellos podrá realizar las validaciones.

5. Obtener los usuarios asignados a un grupo

Para obtener todos los usuarios del grupo <id_grupo>, hay que realizar la siguiente llamada:

- GET **/api/departments/<id_grupo>/users**
- Devuelve un JSON con los usuarios del grupo,

```
{
    "data": [{
        "id": <id_user_1>,
        "id_role”: 3,
    “name”: “Manuel Rodiguez”,
    “email”: “mrodriguez@here.com”
   ...

    },{
        "id": <user_2>,
 "id_role”: 3,
   ...

    },
...
{
        "id": <user_1>,
 "id_role”: 2,
   ...
    }],
...
}
```

6. Obtener los grupos a los que está asignado un usuario

Para obtener todos los grupos a los que está asignado el usuario <id_usuario>, hay que realizar la siguiente llamada:

- GET **/api/users/<id_user>/departments**
- Devuelve un JSON con los grupos del usuario:

```
{
    "data": [{
        "id": <id_group_1>,
        "name”: “Group 1”,
```

```
    ...

     },{
          "id": <id_group_2>,
          "name": "Group 2",
    ...

     },
...
{
          "id": <id_group_3>,
          "name": "Group 3",
    ...
     }],
...
}
```

7. Eliminar usuarios de un grupo

No existe un método específico que permita eliminar al usuario <id_user> del grupo <id_grupo>. Hay que hacer un PUT para actualizar los usuarios del departamento y dejar solo aquellos en que se quiera mantener:

- PUT **/api/departments/<id_grupo>**

```
{
   "ids_users": [8, 2985]
}
```

Si se quieren borrar todos, se haría de la siguiente manera, por tanto:

- PUT **/api/departments/<id_grupo>**

```
{
   "ids_users": []
}
```

2.2.3 Gestión de Workflows

Atención. Estas llamadas son de uso muy restringido, y solo pueden usarse de manera interna o por usuarios autorizados.

1. Crear un workflow

- POST **/api/workflows**

```
{
```

```
"name": "WF Test",
"company_id": "<id_empresa>",
"phases": [
  {
    "status_id": <id_report_status>,
    "transitions": [
      {
        "action": "Iniciar Revisión",
        "status_id": <id_report_status>,
        "responsables": [
          {
            "responsable_id": <id_user>,
            "responsable_type": "App\\Model\\User",
            "sufficient": true
          }
        ]
      }
    ]
  },
  {
    "status_id":<id_report_status>,
    "transitions": [
      {
        "action": "Aprobar",
        "status_id": <id_report_status>,
        "responsables": [
          {
            "responsable_id": <id_role>,
            "responsable_type": "App\\Model\\Role",
            "sufficient": true
          },
          {
            "responsable_id": <id_role>,
            "responsable_type": "App\\Model\\Role",
            "sufficient": false
          }
        ]
      }
    ]
  },
  {
    "status_id": <id_report_status>,
    "transitions": [
      {
        "action": "Tramitar",
        "status_id": <id_report_status>,
        "responsables": [
          {
```

```
                "responsable_id": 2,
                "responsable_type": "App\\Model\\Role",
                "sufficient": 1
              },
              {
                "responsable_id": 3,
                "responsable_type": "App\\Model\\Role",
                "sufficient": 1
              }
            ]
          }
        ]
      },

      {
        "status_id": <id_report_status>,
        "transitions": []
      }
    ]
}
```

- Devuelve un JSON con los datos del workflow creado

```
{
    "data": {
        "name": "WF Test",
        "company_id": "<id_empresa>",
        "initial_phase_id": 498,
        "id": 107
    },
    "status": "ok"
}
```

2. Validar un workflow

Es posible validar la estructura de un workflow antes de crearlo/actualizarlo, para ello hay que realizar una llamada POST con la estructura que se quiera validar:

- POST **/api/workflows/validate**

```
{
   "name": "WF Test",
   "company_id": "<id_empresa>",
   "phases": [
     {
       "status_id": <id_report_status>,
       "transitions": [
         {
```

```
      "action": "Iniciar Revisión",
      "status_id": <id_report_status>,
      "responsables": [
        {
          "responsable_id": <id_user>,
          "responsable_type": "App\\Model\\User",
          "sufficient": true
        }
      ]
    }
  ]
},
{
  "status_id":<id_report_status>,
  "transitions": [
    {
      "action": "Aprobar",
      "status_id": <id_report_status>,
      "responsables": [
        {
          "responsable_id": <id_role>,
          "responsable_type": "App\\Model\\Role",
          "sufficient": true
        },
        {
          "responsable_id": <id_role>,
          "responsable_type": "App\\Model\\Role",
          "sufficient": false
        }
      ]
    }
  ]
},
{
  "status_id": <id_report_status>,
  "transitions": [
    {
      "action": "Tramitar",
      "status_id": <id_report_status>,
      "responsables": [
        {
          "responsable_id": 2,
          "responsable_type": "App\\Model\\Role",
          "sufficient": 1
        },
        {
          "responsable_id": 3,
          "responsable_type": "App\\Model\\Role",
```

```
                "sufficient": 1
              }
            ]
          }
        ]
      },

      {
       "status_id": <id_report_status>,
       "transitions": []
      }
    ]
}
```

- Devuelve un estado 200 - OK en caso que la estructura del workflow sea válida o un JSON con los errores encontrados asociados a los elementos que componen la estructura del workflow (user_id, role_id, status_report_id…, o errores genéricos del workflow (WorkFlow).

```
{
 "message": "The given data was invalid.",
 "errors": {
     "phases": [
         {
             "<user_id>": [
                 "errorResponsableUserId"
             ],
             "<status_report_id>": [
                 "errorDestiantionStatusNoOrigin"
             ],
             "<status_report_id>": [
                 "errorStatusNoInputTransition"
             ],
             "WorkFlow": [
                 "errorNoPath"
             ]
         }
     ]
 }
}
```

Los estados de error devueltos son los siguientes:

- invalidWorkflow: Workflow inválido, no existe o el usuario autenticado no puede usarlo.
- invalidCompanyIdNotExist: Empresa inválida, no existe.
- invalidCompanyIdNotOwer: Empresa inválida, usuario no propietario.
- invalidDepartmentNotExist: Departamento inválido, no existe.

- invalidDepartmentNotOwner: Departamento inválido, el usuario no es propietario.
- invalidUserIdNotExist: Identificador de usuario inválido, no existe.
- invalidUserIdNotOwner: Identificador de usuario inválido, usuario no propietario.
- invalidUserIdNotAssignable: Identificador de usuario inválido, no asignable al workflow.
- errorAttachUserWorkflowUserNotOwner: No se puede adjuntar el usuario, el usuario autenticado no es dueño del workflow.
- errorCreateWorkflowCompany: El usuario autenticado no puede crear un workflow en esta empresa.
- invalidStatusId: Estado inválido, no existe o el usuario autenticado no puede usarlo.
- duplicateInitialStatus: Estado de tipo abierto duplicado.
- statusWithoutOutputTransition: Estado sin transición de salida.
- loopingTransition: Transición de bucle.
- errorResponsableRoleId: Rol inválido, no existe o el usuario autenticado no puede usarlo.
- errorResponsableUserId: Usuario inválido, no existe o el usuario autenticado no puede usarlo.
- errorDuplicateRole: Roles responsables duplicados en la transición.
- errorDuplicateUser: Usuarios responsables duplicados en la transición.
- errorExecuteTransition: No es una transición ejecutable, debe haber por lo menos un usuario/rol de tipo suficiente o dos de tipo insuficiente.
- errorDuplicateStatusIdTransition: Estado duplicado en transición.
- errorDestiantionStatusNoOrigin: Estado de destino sin definición de estado de origen.
- errorStatusNoInputTransition: Estado sin transición de entrada.
- errorNoPath: No hay un camino desde el estado inicial al estado final del workflow.
- errorNoInitialStatus: Es necesario definir un estado de tipo abierto.
- errorNoFinalStatus: Es necesario definir, al menos, un estado de tipo aprobado.

3. Clonar un workflow

Es posible clonar la estructura de un workflow, esto quiere decir que posteriormente será necesario asignar los responsables de cada transición y activarlo en las entidades correspondientes (empresa, grupo de validación y/o usuario). Esta operación solo se permite sobre workflows que no tengan estados personalizados, es decir, que sean todos genéricos de Okticket. El nuevo workflow se creará añadiendo el sufijo _copy al nombre del workflow original.

- POST **/api/workflows/clone**

```
{
   "workflow_id": "<workflow_id>"
}
```

- Devuelve un JSON con los datos del workflow clonado

```
{
    "data": {
        "name": "WF Test_copy",
        "company_id": <company_id>
        "initial_phase_id": 512,
        "id": <workflow_id>
    },
    "status": "ok"
}
```

4. Modificar un workflow

Para modificar un workflow <workflow_id>

- PATCH **/api/workflows/<workflow_id>**

```
{
   "name": "WF Test",
   "company_id": "<id_empresa>",
   "phases": [
     {
       "status_id": <id_report_status>,
       "transitions": [
         {
           "action": "Iniciar Revisión",
           "status_id": <id_report_status>,
           "responsables": [
             {
               "responsable_id": <id_user>,
               "responsable_type": "App\\Model\\User",
               "sufficient": true
             }
           ]
         }
       ]
     },
     {
       "status_id":<id_report_status>,
       "transitions": [
         {
           "action": "Aprobar",
           "status_id": <id_report_status>,
```

```
            "responsables": [
              {
                "responsable_id": <id_role>,
                "responsable_type": "App\\Model\\Role",
                "sufficient": true
              },
              {
                "responsable_id": <id_role>,
                "responsable_type": "App\\Model\\Role",
                "sufficient": false
              }
            ]
          }
        ]
      },
      {
        "status_id": <id_report_status>,
        "transitions": [
          {
            "action": "Tramitar",
            "status_id": <id_report_status>,
            "responsables": [
              {
                "responsable_id": 2,
                "responsable_type": "App\\Model\\Role",
                "sufficient": 1
              },
              {
                "responsable_id": 3,
                "responsable_type": "App\\Model\\Role",
                "sufficient": 1
              }
            ]
          }
        ]
      },

       {
        "status_id": <id_report_status>,
        "transitions": []
      }
    ]
}
```

- Devuelve un JSON con los datos del workflow creado

```
{
    "data": {
```

```
        "name": "WF Test",
        "company_id": "<id_empresa>",
        "initial_phase_id": 498,
        "id": 107
    },
    "status": "ok"
}
```

5. Eliminar un workflow

Para borrar un borrar un <workflow_id> es necesario que no esté en uso ni tenga ningun elemento asignado (Empresa, Grupo de validación y/o usuario)

- DEL **/api/workflows/<workflow_id>**

6. Asignar un workflow

Para activar un workflow es necesario asignarlo a los diferentes elementos que hacen uso de él:

1. Empresa
2. Grupo de validación
3. Usuario

- PATCH **/api/workflows/<workflow_id>**

```
{
   "ids_users_add": [<user_id>,<user_id>,...],
   "ids_users_del": [<user_id>,<user_id>,...],
   "ids_departments_add": [<department_id>,<department_id>,...],
   "ids_departments_del": [<department_id>,<department_id>,...],
   "ids_companies_add": [<company_id>,<company_id>,...],
   "ids_companies_del": [<company_id>,<company_id>,...]
}
```

Se comprueba que el workflow sea válido antes de poder asignar/desasignar elementos y se comprueba que el usuario tenga permiso sobre los diferentes elementos que se intentan asignar.

- Devuelve un JSON con los datos del workflow activado

```
{
    "data": {
        "id": <workflow_id>,
        "name": "WF Test",
        "initial_phase_id": 508,
        "company_id": <company_id>
    },
```

```
    "status": "ok"
}
```

Los posibles errores de validación de esta operación:

- errorAttachCompanyWorkflowInvalidWorkflow: No es posible asignar una compañia a un workflow que no sea válido.
- errorAttachCompanyWorkflowUserNotOwner: No se puede adjuntar la empresa, el usuario autenticado no es dueño del workflow,,
- errorAttachCompanyWorkflowCompanyNotOwner: No se puede adjuntar la compañía, la compañía no es propietaria,
- errorDetachCompanyWorkflowUserNotOwner: No se puede separar la compañía, el usuario autenticado no es dueño del workflow,
- errorAttachDepartmentWorkflowInvalidWorkflow: No es posible asignar un departamento a un workflow que no sea válido.
- errorAttachDepartmentWorkflowUserNotOwner: No se puede adjuntar el departamento, el usuario autenticado no es dueño del workflow,
- errorAttachDepartmentWorkflowCompanyNotOwner: No se puede adjuntar el departamento, la empresa no es dueña del workflow,
- errorDetachDepartmentWorkflowUserNotOwner: No se puede separar el departamento, el usuario autenticado no es dueño del workflow,
- errorAttachUserWorkflowInvalidWorkflow: No es posible asignar un usuario a un workflow que no es válido.
- errorAttachUserWorkflowUserNotOwner: No se puede asignar el usuario, el usuario autenticado no es dueño del workflow.
- errorDetachUserWorkflowNotOwner: No se puede separar el usuario, el usuario autenticado no es dueño del workflow.
- errorDetachUserWorkflowUserNotOwner: "No se puede separar el usuario, el usuario autenticado no es dueño del usuario.

7. Obtener detalle de un workflow

Para obtener el detalle del workflow <workflow_id>, hay que realizar la siguiente llamada:

- GET **/api/workflows/<workflow_id>/?with=phases.transitions.responsables**
- Devuelve un JSON con la estructura del workflow:

```
{
    "data": {
        "id": <workflow_id>,
        "name": "WF Test",
        "initial_phase_id": 508,
        "company_id": <company_id>,
        "phases": [
            {
                "id": 508,
                "phase": "Fase0",
```

```
    "workflow_id": 106,
    "status_id": 0,
    "transitions": [
        {
            "id": 608,
            "action": "Iniciar Revision (TO)",
            "phase_id": 508,
            "status_id": 2,
            "responsables": [
                {
                    "id": 1326,
                    "transition_id": 608,
                    "responsable_id": 3,
                    "responsable_type": "App\\Model\\Role",
                    "sufficient": 1,
                    "condition": null
                }
            ]
        }
    ]
},
{
    "id": 509,
    "phase": "Fase1",
    "workflow_id": 106,
    "status_id": 2,
    "transitions": [
        {
            "id": 609,
            "action": "Aprovar (TO)",
            "phase_id": 509,
            "status_id": 5,
            "responsables": [
                {
                    "id": 1327,
                    "transition_id": 609,
                    "responsable_id": 3,
                    "responsable_type": "App\\Model\\Role",
                    "sufficient": 1,
                    "condition": null
                },
                {
                    "id": 1328,
                    "transition_id": 609,
                    "responsable_id": 72,
                    "responsable_type": "App\\Model\\Role",
                    "sufficient": 1,
                    "condition": null
```

```
                            }
                        ]
                    }
                ]
            },
            {
                "id": 510,
                "phase": "Fase2",
                "workflow_id": 106,
                "status_id": 4,
                "transitions": []
            },
            {
                "id": 511,
                "phase": "Fase3",
                "workflow_id": 106,
                "status_id": 5,
                "transitions": [
                    {
                        "id": 610,
                        "action": "Tramitar (TO)",
                        "phase_id": 511,
                        "status_id": 4,
                        "responsables": [
                            {
                                "id": 1329,
                                "transition_id": 610,
                                "responsable_id": 2,
                                "responsable_type": "App\\Model\\Role",
                                "sufficient": 1,
                                "condition": null
                            },
                            {
                                "id": 1330,
                                "transition_id": 610,
                                "responsable_id": 3,
                                "responsable_type": "App\\Model\\Role",
                                "sufficient": 1,
                                "condition": null
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "status": "ok"
}
```

8. Obtener asignaciones de un workflow

Para obtener las asignaciones del workflow <workflow_id>, hay que realizar la siguiente llamada:

- GET **/api/workflows/<workflow_id>/?with=companies,departments,users**
- Devuelve un JSON con la estructura del workflow:

```
{
 "data": {
     "id": <workflow_id>,
     "name": "WF Test",
     "initial_phase_id": 508,
     "company_id": <company_id>,
     "companies": [],
     "departments": [],
     "users": [],
 },
 "status": "ok"
}
```